Universität Mannheim
Fakultät für Mathematik und Informatik
Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg

Hauptdiplomklausur Informatik

September1997: Rechnernetze-Praktikum

Name: ..... Vorname: .....

Matrikel-Nr.: ..... Semester: ..... Fach: .....

Hinweise:

(a) Bitte füllen Sie sofort den Kopf des Deckblatts aus.
(b) Überprüfen Sie Ihr Klausurexemplar auf Vollständigkeit (10 Seiten).
(c) Tragen Sie Ihre Lösungen soweit möglich direkt in die Klausur ein.
(d) Es sind keine Hilfsmittel zugelassen
(e) Zeit: 67 Minuten

Aufgabe	max. Punktezahl	Punkte
1	25	
2	25	
3	17	
Gesamt	67	

Aufgabe 1 [2 + 4 + 7 + 12 = 25 Punkte]: Java

(a) [2 Punkte] Java unterstützt das Thread-Konzept. Was versteht man unter einem Thread?

(b) [4 Punkte] In Java kann man eigene Threads durch Vererbung oder durch die Implementierung von Interfaces realisieren. Erklären Sie kurz, was man unter Vererbung und der Implementierung eines Interfaces versteht.

(c) [7 Punkte] Das folgende Code-Fragment führt auf einigen Plattformen zu einem Deadlock. Erläutern Sie ausführlich, warum es zu diesem Fehler kommt! Gehen Sie bei Ihren Überlegungen davon aus, daß der Java-Code auf einer Plattform abläuft, die “time-slice multitasking” unterstützt. (D.h. der Prozessor wird an jeden Thread für einen festgelegten Zeitschlitz vergeben. Am Ende eines Zeitschlitzes wird dem Thread der Prozessor wieder entzogen und an den nächsten Thread vergeben.)

Erläuterung zum Schüsselwort `synchronized`: Vor dem Ausführen einer `synchronized` Methode auf einem Objekt, wird eine Sperre auf das Objekt gesetzt. Andere Threads, die ebenfalls eine `synchronized` Methode auf demselben Objekt ausführen wollen, können das Objekt erst dann für sich sperren, wenn die Sperre durch den ersten Thread wieder aufgehoben ist.

```
class Array {
    private int[] a = new int[100000];

    public synchronized int DivSums(Array pa) {
        return Sum() / pa.Sum();
    }
    public synchronized int Sum() {
        int s = 0;
        for (int j = 0; j < a.length; j++) { s = s + a[j]; }
        return s;
    }
}

class DivSumThread extends Thread {
    Array ta1, ta2;
    int div = 0;

    public DivSumThread(Array pa1, Array pa2) {
        ta1 = pa1;
        ta2 = pa2;
    }
    public void run() {
        div = ta1.DivSums(ta2);
    }
}

public class AppObject extends Applet {
    Array a1 = new Array();
    Array a2 = new Array();
    DivSumThread t1 = new DivSumThread(a1, a2);
    DivSumThread t2 = new DivSumThread(a2, a1);

    public void init() {
        /* Initialize Arrays */
        t1.start();
        t2.start();
    }
}
```

(d) [12 Punkte] Modifizieren Sie nun das Java-Code-Fragment so, daß ein Deadlock nicht mehr auftreten kann. Verändern Sie hierbei die Semantik des Programms nicht. Insbesondere soll die Klasse Array nicht verändert werden. Erläutern Sie zunächst verbal, wie Sie den Deadlock verhindern können. (Gehen Sie wie in Aufgabenteil (c) davon aus, daß der Java-Code auf einer Plattform läuft, die “time-slice multitasking” unterstützt.)

Hinweis: Sie können für Ihre Lösung eine zusätzliche Klasse einführen.

Aufgabe 2 [4 + 6 + 9 + 6 = 25 Punkte]: TCP/IP

(a) [4 Punkte] Nennen Sie die wichtigsten Unterschiede zwischen TCP und UDP.

(b) [6 Punkte] In einer multimedialen Anwendung in Client/Server-Architektur könnte beispielsweise der folgende Bedarf zur Übertragung von Daten bestehen:

- die Übertragung von Audioclips in Echtzeit,
- die periodische, zeitunkritische Übertragung von Bitmap-Bildern und
- die ständige Übertragung von kurzen Kontrolldaten zur Steuerung der Anwendung.

Welches der beiden Transportprotokolle TCP und UDP wird sinnvollerweise für welche der genannten Daten verwendet? Begründen Sie Ihre Entscheidung jeweils.

(c) [9 Punkte] Sie möchten mit Hilfe von Sockets einen Spiele-Server entwickeln. Ihr Server soll mehrere Clients gleichzeitig bedienen können, damit mehrere Spieler gegeneinander spielen können. Zwischen einem Client und dem Server soll während der gesamten Spieldauer eine Verbindung bestehen.

Schreiben Sie die für den Verbindungsauf- und -abbau nötigen Programmteile in Pseudocode. Entwickeln Sie eine Server- und eine Client-Implementation. Erzeugen Sie im Server pro Client jeweils einen Prozeß und verwenden Sie einen Prozeß, um auf eingehende Verbindungswünsche zu warten. Zum Zugriff auf Sockets stehen die aus der Programmiersprache C bekannten Bibliotheksfunktionen zur Verfügung.

(d) [6 Punkte] Erläutern Sie die Funktion der von Ihnen in Aufgabenteil (c) verwendeten Socket-Bibliotheksfunktionen.

Aufgabe 3 [9 + 4 + 4 = 17 Punkte]: RPC

(a) [9 Punkte] Skizzieren Sie den grundsätzlichen Ablauf eines RPC Prozeduraufrufs im Internet. Erklären Sie dabei auch den Begriff XDR und zeigen Sie, wo in Ihrer Ablaufskizze im Zusammenhang mit XDR benötigte Operationen ausgeführt werden.

(b) [4 Punkte] Nennen Sie Fehlerquellen, die im Ablauf eines RPC Prozeduraufrufs (im Gegensatz zum lokalen Prozeduraufruf) auftreten können.

(c) [4 Punkte] Beschreiben Sie zwei Semantiken für RPC Prozeduraufrufe.